AMENDMENTS TO THE SPECFICIATION

Please delete the paragraph beginning on page 6, line 23 and ending on page 6, line 29.

Please delete the paragraph beginning on page 7, line 1 and ending on page 7, line 14.

Please amend the paragraph beginning on page 4, line 20 as follows:

Often, the process of configuring a shared resource such as a PDU results in the bringing in of a component by a subsystem in one product line (such as the first server) into a second subsystem in a second product line (such as the second server). The instantiation of the component typically is not a problem because the component probably falls into both product lines. But constraints on the installation of this component common to both product lines may generate subsequent requires_component decision points that must be satisfied by the components that fall exclusively within the product line of the second subsystem. Thus, unless all components from all product lines are available, satisfaction of these subsequent requires_component statements may [[be]] include incompatible components. Of course as previously discussed, failing to restrain the choice of components to one product line context makes configuration of these heterogeneous systems arduous.

Please amend the paragraph beginning on page 5, line 2 as follows:

The heterogeneous configurator of the invention employs a process by which the configurator may configure components that span several contexts, such as product lines without undo burden being placed on a model used to represent the components. The components are represented in the model as a class called component. The class includes a constraint that requires each instantiated object to determine whether the context that is appropriate to the component object is the same as the current context state for the configuration. If the answer is no, the installation of the object changes the current state of the configuration to the state appropriate to it. The appropriate state may be associated with the object component in a number of ways, including an attribute associated with the object or by the context subsystem in

which it resides. Each time a decision point statement such as a requires_component is encountered by the configuration engine, the current state of the context of the configuration is cached locally just in case the installation of the required object changes the current state of the context. Once installation of the object is completed, if its installation resulted in a change of the context state, the cached context is restored before proceeding to the next step in the configuration. If the installation of the object generates additional decision point statements, ~~there~~ their execution is processed in the same way. Thus, if any of them results in a change of state, the changes in state are nested and are restored as the installation of their object components are completed.

Please amend the paragraph beginning on page 8, line 5 as follows:

One embodiment of the heterogeneous configurator method and apparatus of the invention includes a model representing a plurality of components, each of the components spanning two or more contexts such as two or more product lines. The components are organized into a context hierarchy, such as a product line hierarchy. The model can be coded using a configuration modeling language that establishes decision points through statements such as requires_component or "demand" or like statements appurtenant to the particular language used. The model provides a context attribute for each of the components. The model may also contemplate the creation of subsystems which provides localized segmentation of components in a configuration by which particular groups of components are segregated from others as being part of a particular portion of the entire system, and which can be limited to a single context. The model includes a constraint introduced at the highest level of the class hierarchy for the class component. This constraint requires an instantiated object of the class component to always determine what its context (e.g. product line) should be. The execution of this constraint can be accomplished using [[a]] the context of a subsystem to which the object belongs as an indication of its appropriate context, or it may be some attribute associated directly with the component.